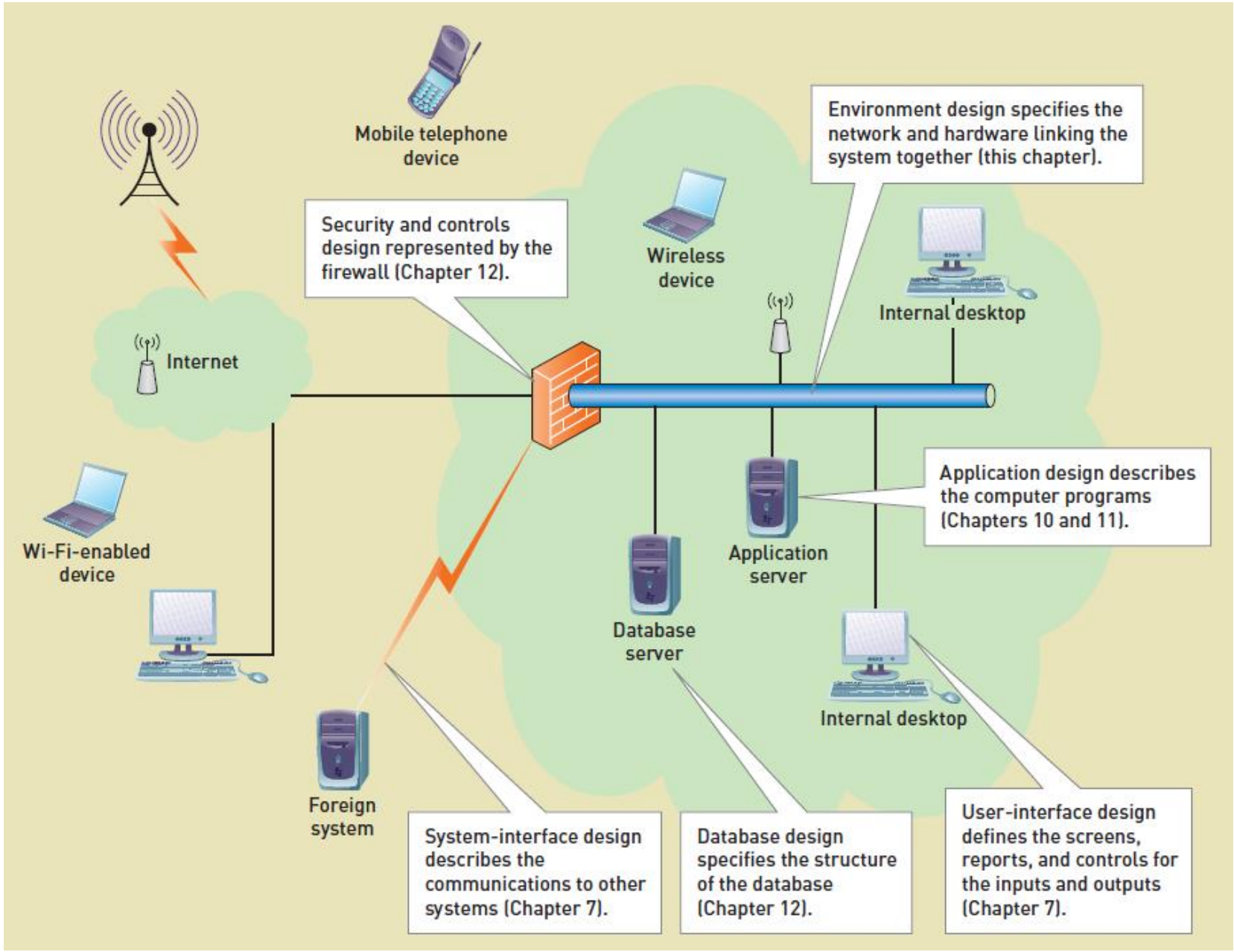
Inxhinieria **Softuerike**

Faza: Dizajnit apo Modelimit te Sistemit

Ramiz HOXHA
ramiz.hoxha@ubt-uni.net
2020/2021

Komponentet/elementet e një sistemi softuerik



Rikujtim: Baza e zgjidhjes

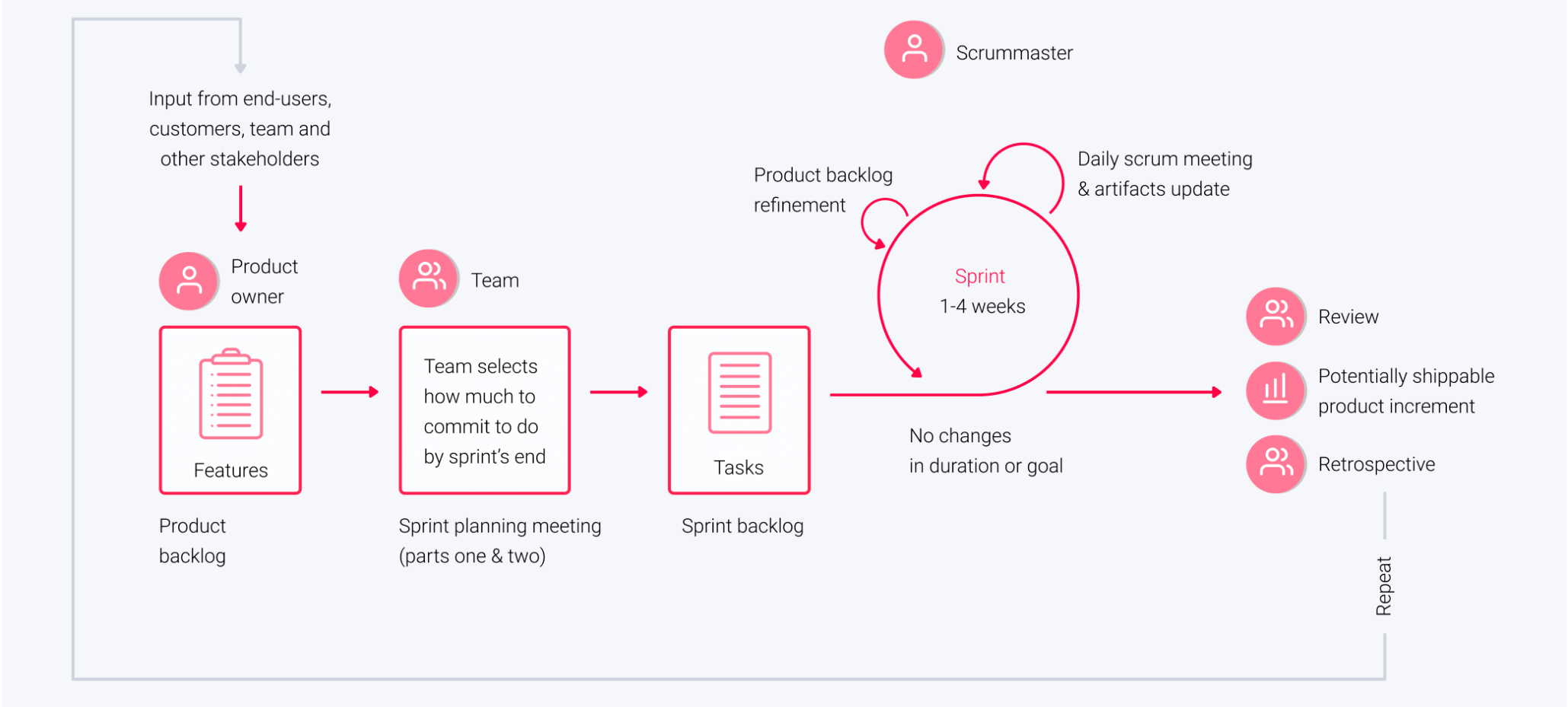
- Kërkesat -> Dizajni -> Implementimi

- Në Analizën e Sistemeve ne kuptojmë…
  - **Çfarë** është nevojat e biznesit

- Në Modelimin/Dizajnimi i Sistemit ne kuptojmë…
  - **"Si"** sistemi do të ***konfigurohet*** dhe ***ndërtohet*** që të plotëson ato nevoja

- E gjithë puna "logjike" nga *analiza e sistemit* është konvertuar në "fizike"

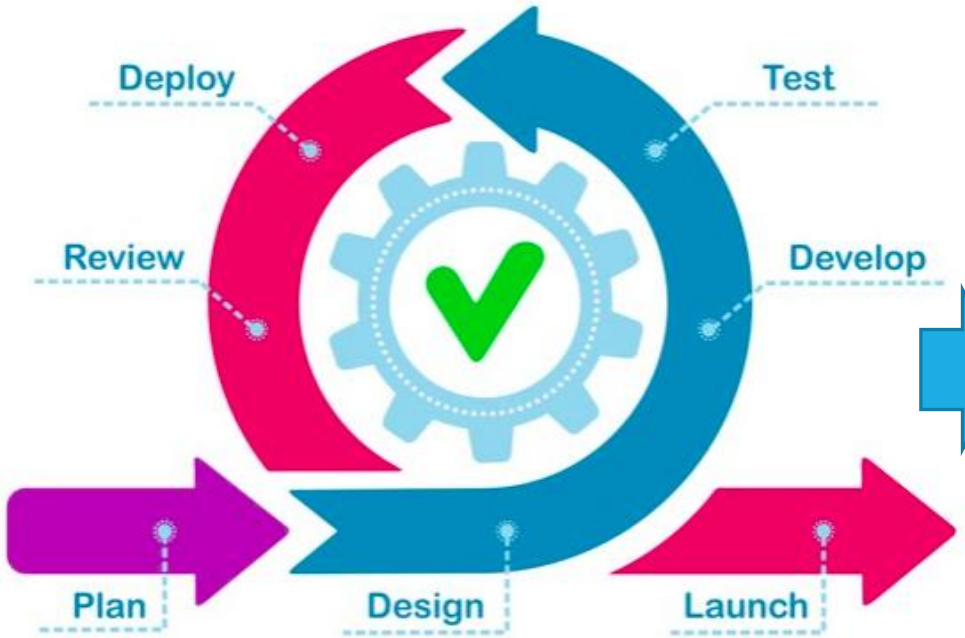
Rikujtim: Qasja e SCRUM'it

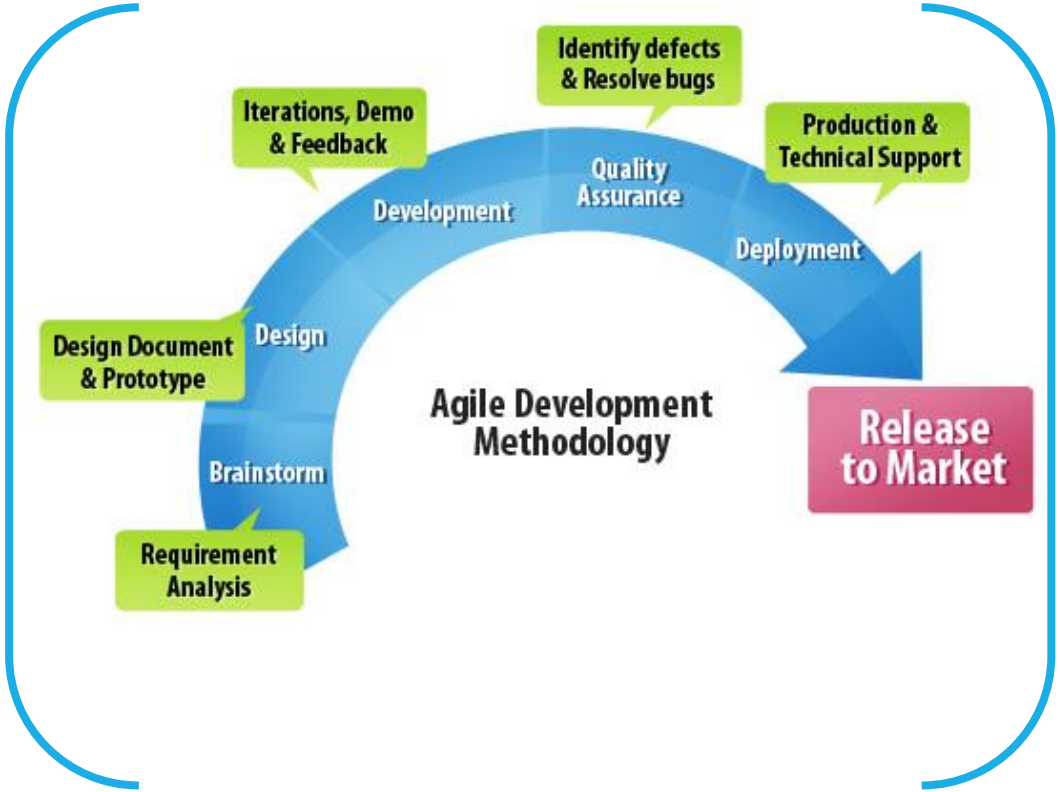


Rikujtim: Cikli i Jetës së Zhvillimit të Softuerit

Fazat e SDLC:

1. Analiza
2. Dizajni
3. Implementimi
4. Testimi
5. Dorzimi/Zbatimi
6. Mirmbajtja

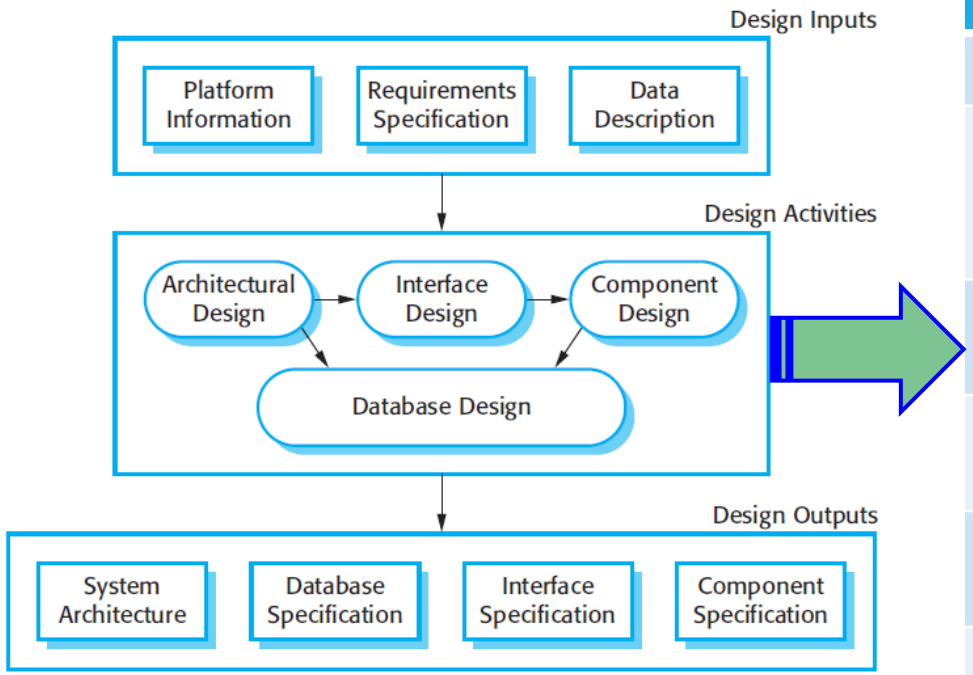




This model represents the SDLC applied to an agile process

Faza e Dizajnit

Aktiviteti i Dizajnit:



Aktiviteti i Dizajnit	Elementet ose Çështje
Dizajni i rrethinës ose mjedisit	Pajisjet, Rrjetet, Serverët
Dizajni i arkitektures së aplikacionit dhe softuerit/programit	Softueri (module, komponenteve, etj) dhe Konfikurimet
Dizajni i interfasës së sistemit	Mjeti për komunikim (API) me sisteme tjera
Dizajni i ndërfaqeve të përdoruesve	Dritarja e përdoruesit , Raportet
Dizajni i bazës së të dhënave	Magazinimi i informacioneve apo të dhënave
Dizajni i kontrollit qasjes dhe Sigurës së sistemit	Firewalls, Qasjet/Kontrolli qasjes

Faza e Dizajnit...

- Faza e dizajnit është kur inxhinierët e softuerit identifikojnë komponentet teknikë të nevojshëm për të përmbushur kërkesat.
  - **Ekipi** vendos elemente të tilla si mënyra se si softueri i ri **integrohet në arkitekturën** e përgjithshme të një sistemi, **sekemat e bazës së të dhënave** të nevojshme për të vendosur të dhëna të reja, dizajnin e **ndërfaqes/interfasave** së përdoruesit dhe një mori detajesh të tjera teknike.
  - **Inxhinierët** rishikojnë (review) tregimet e përdoruesve me **pronarin e produktit** për të kuptuar nevojat, pastaj ata **nxjerrin apo definojn** komponentet teknikë.
- Kjo fazë përfshin procesin e dizajnimit të elementeve të një sistemi siç janë arkitektura, modulet dhe komponentet, ndërfaqet/interfasat (GUI dhe komunikimit/API) e ndryshme të atyre komponenteve dhe të dhënat që kalojnë nëpër atë sistem.

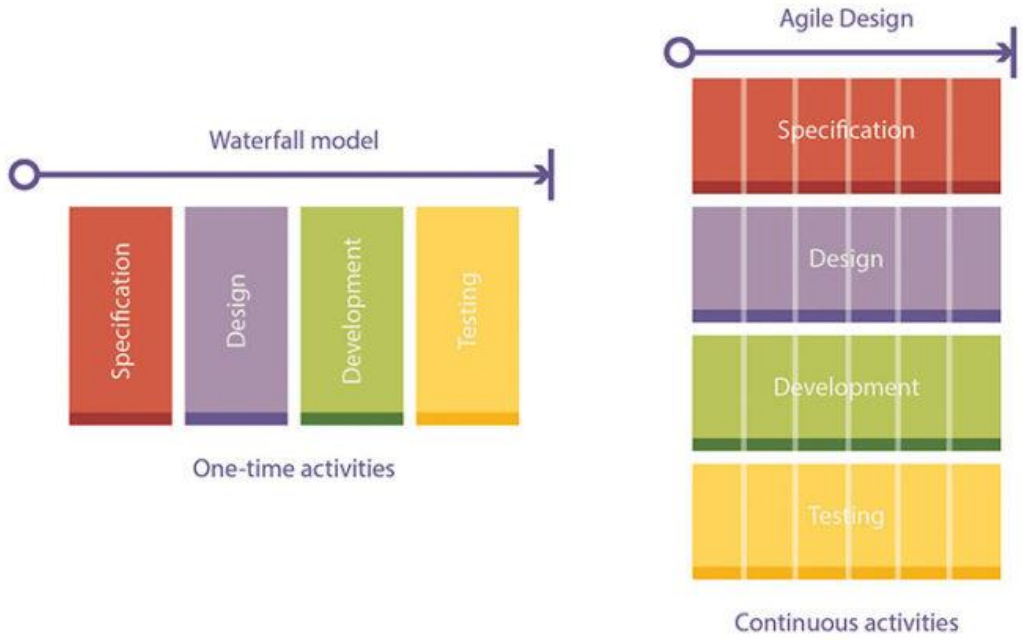
Faza e Dizajnit...

- Elemente që duhët dizajnohen në këtë faze:
  - **Arkitektura** – Ky është modeli konceptual që përcakton strukturën, sjelljen dhe më shumë pamje të një sistemi. Ne mund të përdorim diagrame për të reprezentuar dhe ilustruar arkitekturën.
  - **Modulet** – Këto janë komponentet që trajtojnë "**një detyrë/operacion specifike**" në një sistem. Kombinimi i moduleve e përbëjnë sistemin.
  - **Komponentët** – Ky ofron një funksion ose grup të veçantë të funksioneve të nderlidhuar. Këto funksione përbëhen nga module.
  - **Ndërfaqet** – Ky është kufiri i përbashkët përmes së cilës komponentët e një sistemi shkëmbejnë informacionin dhe nderlidhen.
  - **Baza e të dhënat** – Ky është menaxhimi i informacionit dhe rrjedha e të dhënave.

Krijoni dizajnin në Agile

- Në një mjedis Agile fazat ecin paralelisht në vend që të ndjekin njëra-tjetrën.
  - Ne dizajnojmë, zhvillojmë dhe testojmë në të njëjtën kohë.
  - Ne e ndajmë produktin në pjesë më të vogla, të pavarura dhe të zbatueshme, të cilat mund të dorzohen individualisht.

- Një proces Agile i dizajnit lejon që të përdoret qasja
  - Qasje **Iterative** (përsëritëse) si dhe
  - Qasje **Incremental** (rritje) për të ofruar dizajn për klientin tonë.

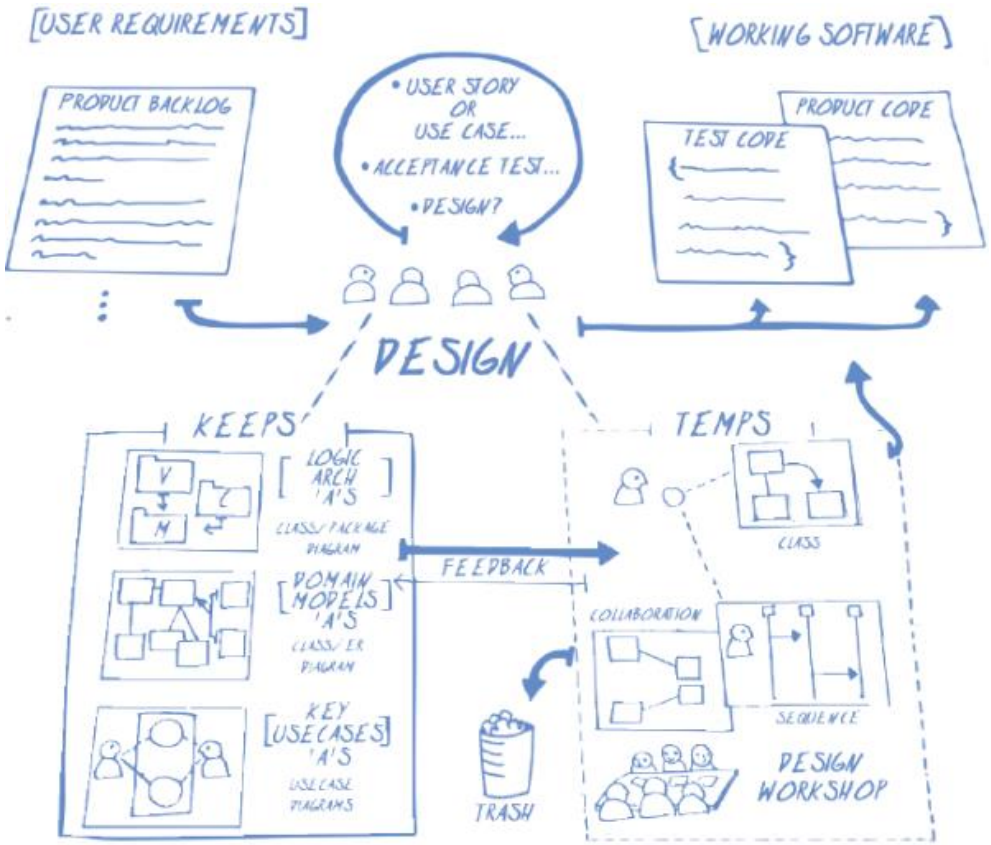


Krijoni dizajnin në Agile...

- Rekomandimi i modeleve "Big Picture" për tu mbajtur dhe mirëmbajtur konsiston në:
  - "**Arkitektura**" e sistemit që ekipi të ketë një ide të përafërt të të gjithë strukturës së sistemit.
    - Arkitektura si: Diagrame të Package, Component ose Deployment
  - "**Modeli i domenit**" për të ndihmuar ekipin të kuptojë konceptet e përdorura në domenin e aplikacionit.
    - Modeli i domenit si: Diagramin e moduleve domenit dhe apo ose Diagramet e klasës
  - "**Use Case kryesore**" për të kuptuar përdoruesit tipikë të sistemit, dhe mënyrën se si ata përfitojnë nga sistemi.
    - Use Case kryesore si: Diagramet e Use Cas'ave + Diagramat e Sequences / Komunikimit

Krijoni dizajnin në Agile...

- Çdo **vendim i dizajnit** që merrni duhet të **drejtohet apo shtyrë (driven)** nga kërkesat
- **funksionale** dhe **jofunksionale**.



Procesit të Dizajnit të Softuerit

1. Dizajni duhet të ekspozojë një patern arkitektektonike të njohur si:
   - (MVC, Layer, Hexagonal, Microservice etj)
     - komponimi i komponenteve me karakteristika të mira në dizajn dhe
     - implementimi në mënyre evolucionare, duke faselituar testimin dhe dorzimi.
2. Dizajni i modelit të domenit të jetë modular:
   - softueri duhet të jetë i ndarë në mënyrë logjike në nën-domene ose nën-sisteme.
3. Dizajn duhët të përmbaj në form të theksuar apo të veqant:
   - **bazes së të dhënave**, **arkitekturës**, **interfaces**, dhe **komponentëve** (moduleve).
4. Dizajn duhët të orientohet:
   - në **strukturim** të të dhënave të përshtatshme për **Klasta** që do të implementohe dhe që janë të **varura nga paternat** të njohura të të dhënave.
   - në **komponente** që shfaqin ose ekspozojnë karakteristika **funksionale të pavarura**.
   - në **ndërfaqet (interfaces ose API'ja)** që **thjeshtzojn kompleksitetin e lidhjeve** midis **komponentëve** me mjedisin e jashtëm.

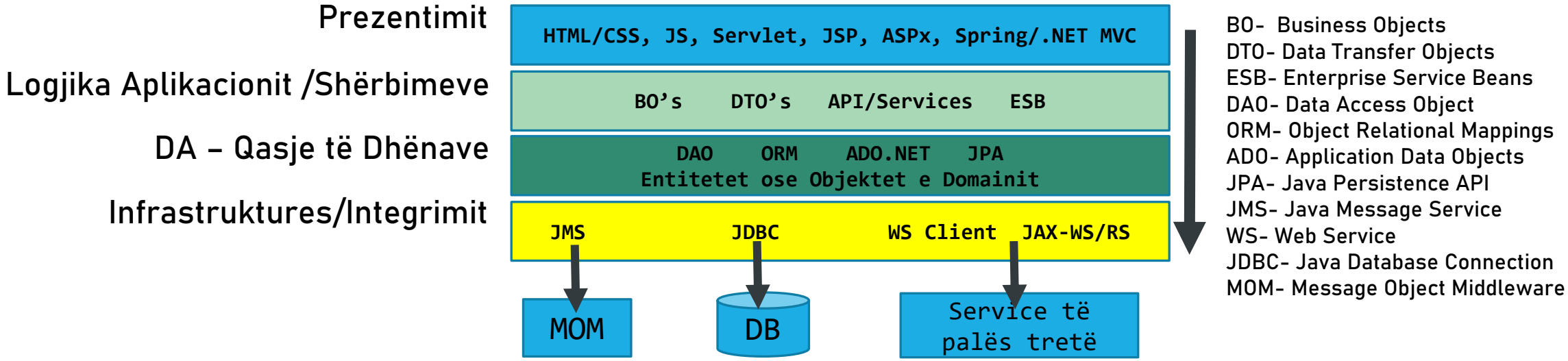
Procesit të Dizajnit të Softuerit…

5. Një dizajn duhet të rrjedhin:
   - duke përdorur një metodë përseritse+rritës që është nxitur nga **informacionet** e marra gjatë **analizës kërkesave** të softuerit.
6. Një dizajn duhet të reprezentohet duke përdorur:
   - **simbolet** që në mënyrë efektive dhe të *qartë komunikojn* përmbajtjen e tyre.

Paternat: Arkitektures softuerike

Arkitekture shtresore (layer):

- **Komponentët** e arkitekturës **shtresore** janë të **organizuara** në **shtresa horizontale** fig më poshtë,
- Secila **shtresë** që **kryen një rol** apo **përgjegjsi specifik** brenda aplikacionit.
- **Paterna** nuk specifikon **numrin** dhe **llojet e shtresave** që duhet të **ekzistojnë** në të,
- Standarte i ka **4-Shtresa**:
  - Prezantimi, Aplikacionit (biznesi/service) , Qasjes të Dhënave, Infrastuktures/integrimit



Shembull: arkitekrures n-shtresore

- ✓ Mund të përmbajë shtresa **shtesë** të hapura, si një shtresë e **shërbimit,** që mund të **përdoren për t'u qasur** shërbimeve e mbrenda shtresës së biznesit.

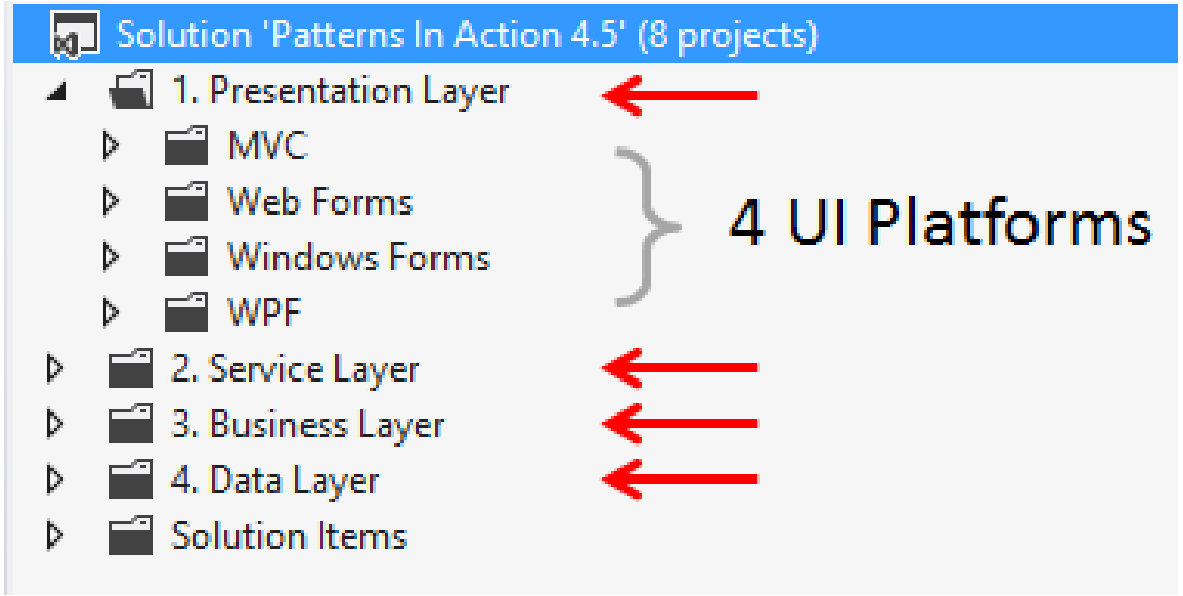
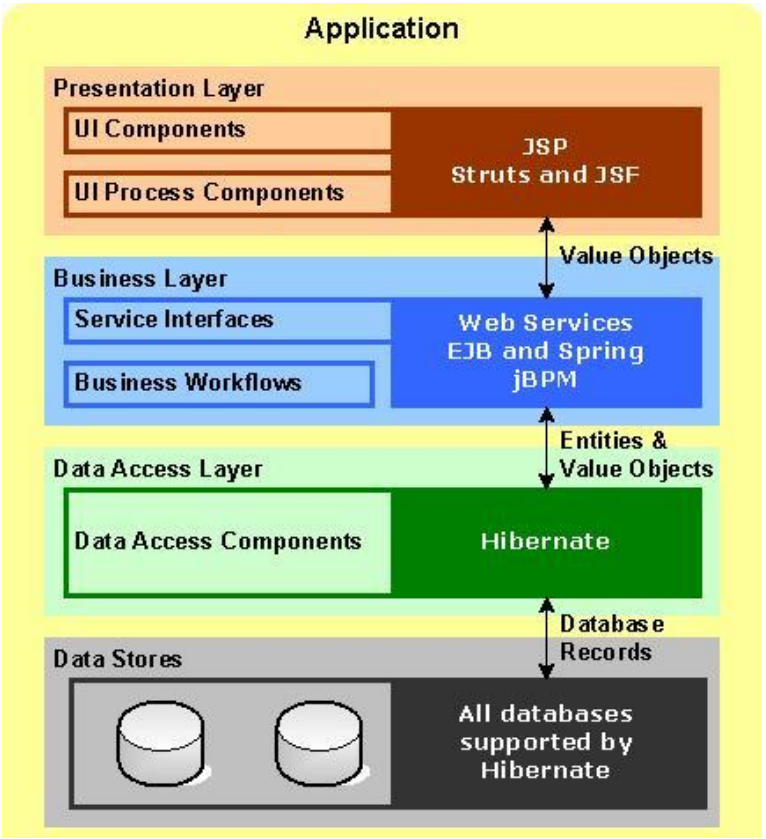


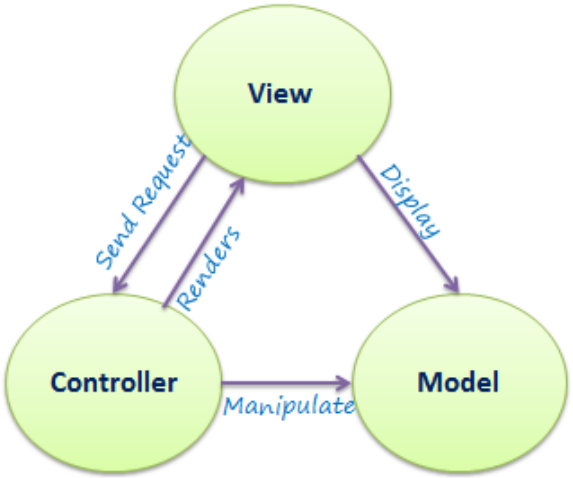
fig organizimi i kodit të aplikacionit n-shtresore/nivel

fig Arkitektura **n**-shtresore/nivel me shtresen e shërbimeve



Paternat: Arkitektures softuerike…

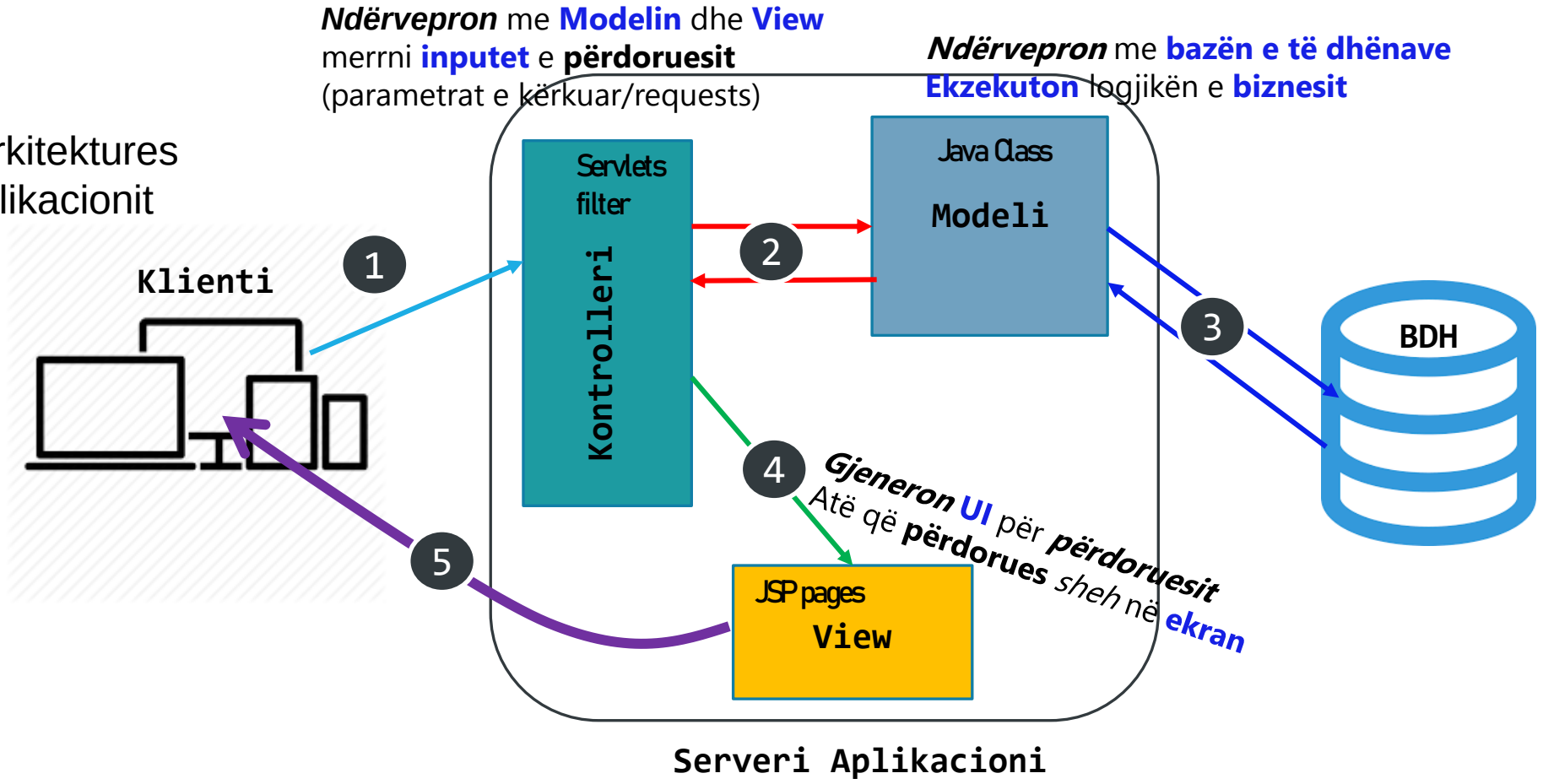
- Arkitektura MVC.
  - MVC qëndron për **M**odel **V**iew **C**ontroller
- Frameworka MVC është një patern arkitektonik që *ndan* aplikacion në *tre komponentë* logjikë kryesorë
  - **Model**, **View** dhe **Controller**
- Secili komponent i arkitektures është *ndërtuar* të trajtoj specifikisht aspektet e implementimit të një aplikacioni



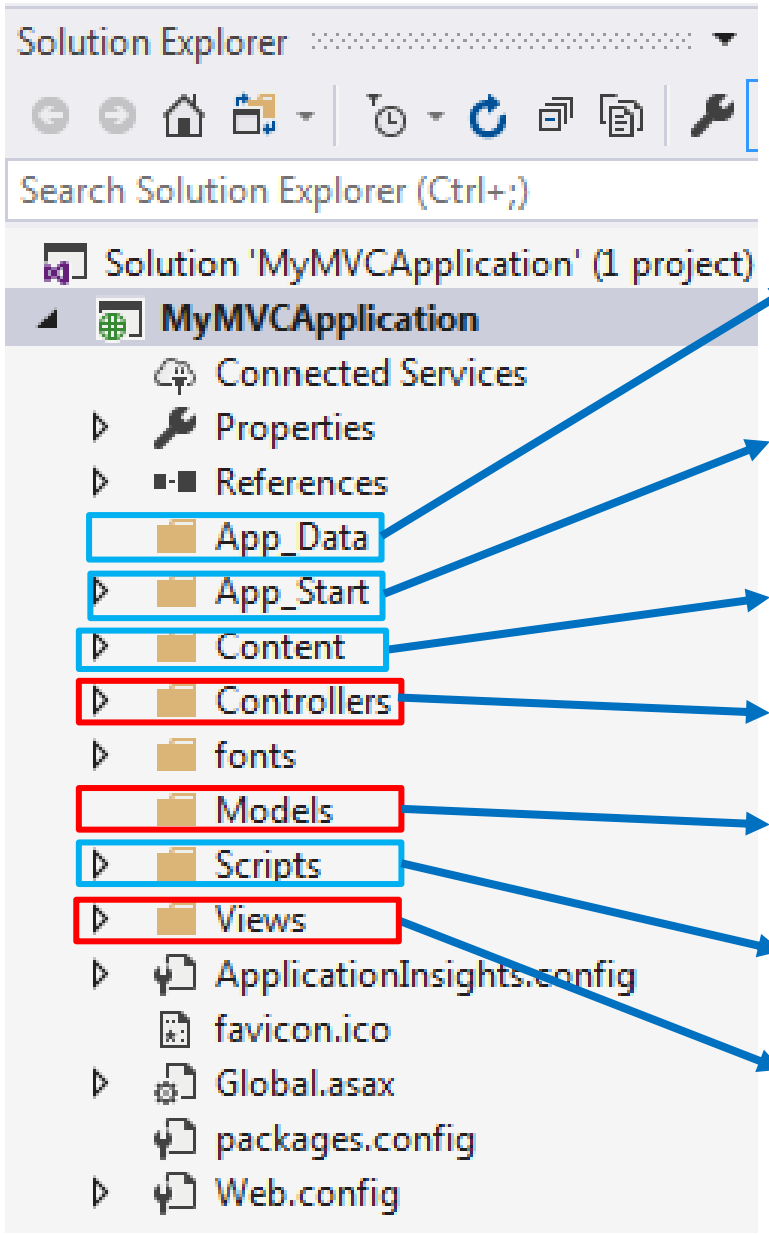
- MVC ndan *logjikën e biznesit* dhe *shtresën e prezantimit* nga njëra-tjetra.
- Arkitektura MVC është *bërë e njohur* për dizajnimi e Ueb aplikacioneve.

Paternat: Arkitektures softuerik...

Paterna e Arkitektures **MVC** të Aplikacionit



Struktura e organizimit të MVC



App_Data – mund të përmbajë **skedarë** me të dhënave të aplikacionit si fila (**LocalDB**, **mdf**, **xml**) dhe filja të tjerë që lidhen me të dhënat. **IIS** kurrë nuk do të ***shërbejë*** fijla nga folderat e **App_Data**.

App_Start folder mund të përmbajë file të **Klasave** të cilat do të ekzekutohen kur fillon aplikacioni. fajlat e konfigurimit si AuthConfig.cs, BundleConfig.cs, FilterConfig.cs, **RouteConfig.cs** etj. MVC 5 përfshin BundleConfig.cs, FilterConfig.cs dhe RouteConfig.cs si **default**.

Content folder përmban fajla statikë si fajla të CSS, imazhe dhe fajla ikonash. Aplikacioni MVC 5 përfshin bootstrap.css, bootstrap.min.css dhe Site.css si default.

Controllers folder përmban klas fijla për kontrollerin. Kontrollorët merren me ***kërkesën (requests)*** e përdoruesve dhe kthen një ***përgjigje (respons)***.

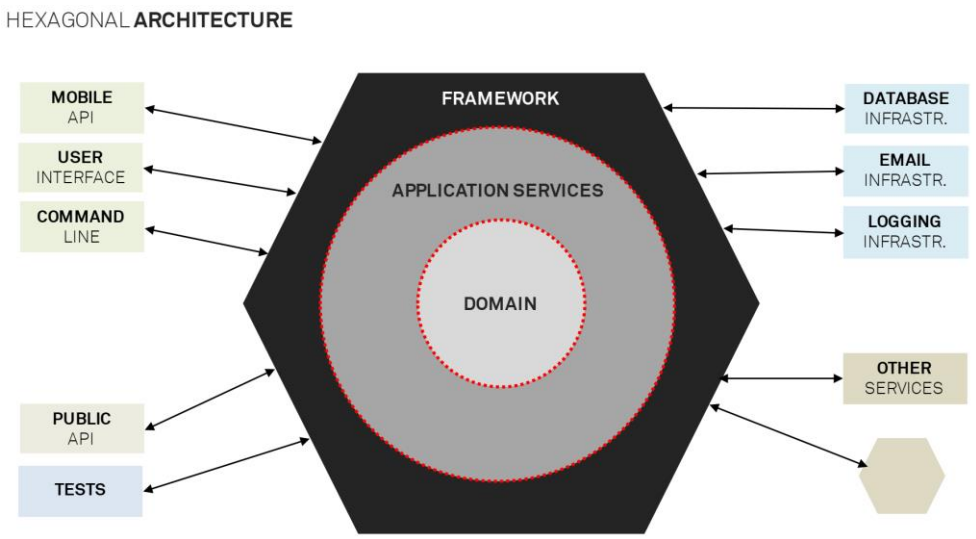
Models folder përmban fijla të klasave të BO. Në mënyrë tipike, klasa e modelit përfshin vetit (properties) publike, të cilat do të përdoren nga aplikacioni për të mbajtur dhe manipuluar të dhënat e aplikacionit.

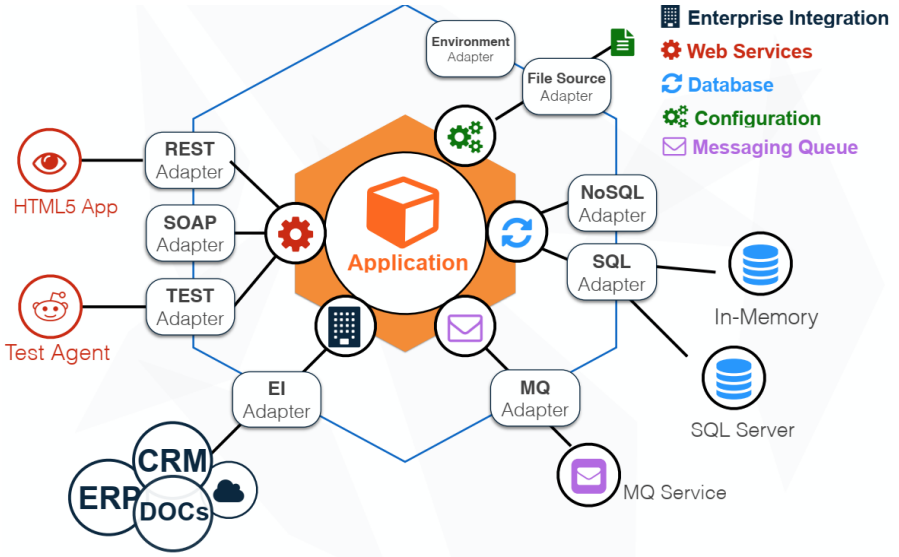
Scripts folder contains fajlat e JavaScript ose Angular për aplikacionin. MVC 5 përfshin fajla të javascript për bootstrap, jquery 1.10 dhe modernizues si default.

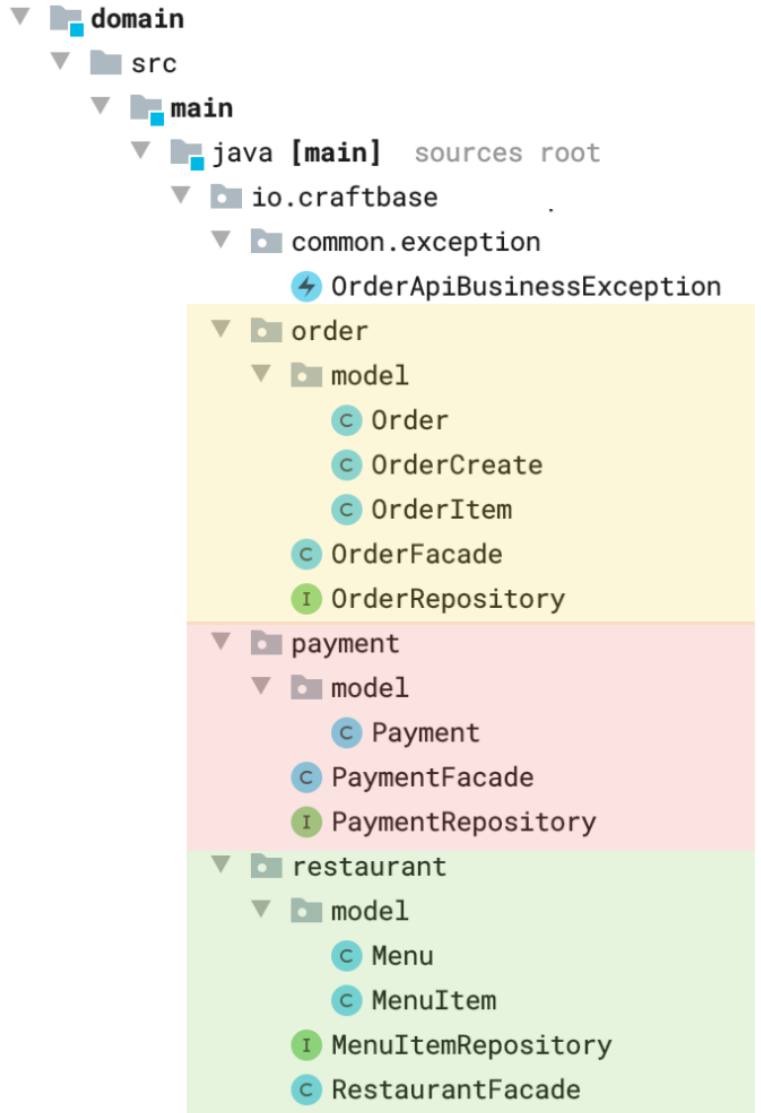
Views folder përmban fajla html për aplikacionin. Tipik VIEW fajlat është një fajl .cshtml ku ju krijoni html dhe HTML/CSS, JS, Servlet, JSP, ASPx, Spring/.NET MVC

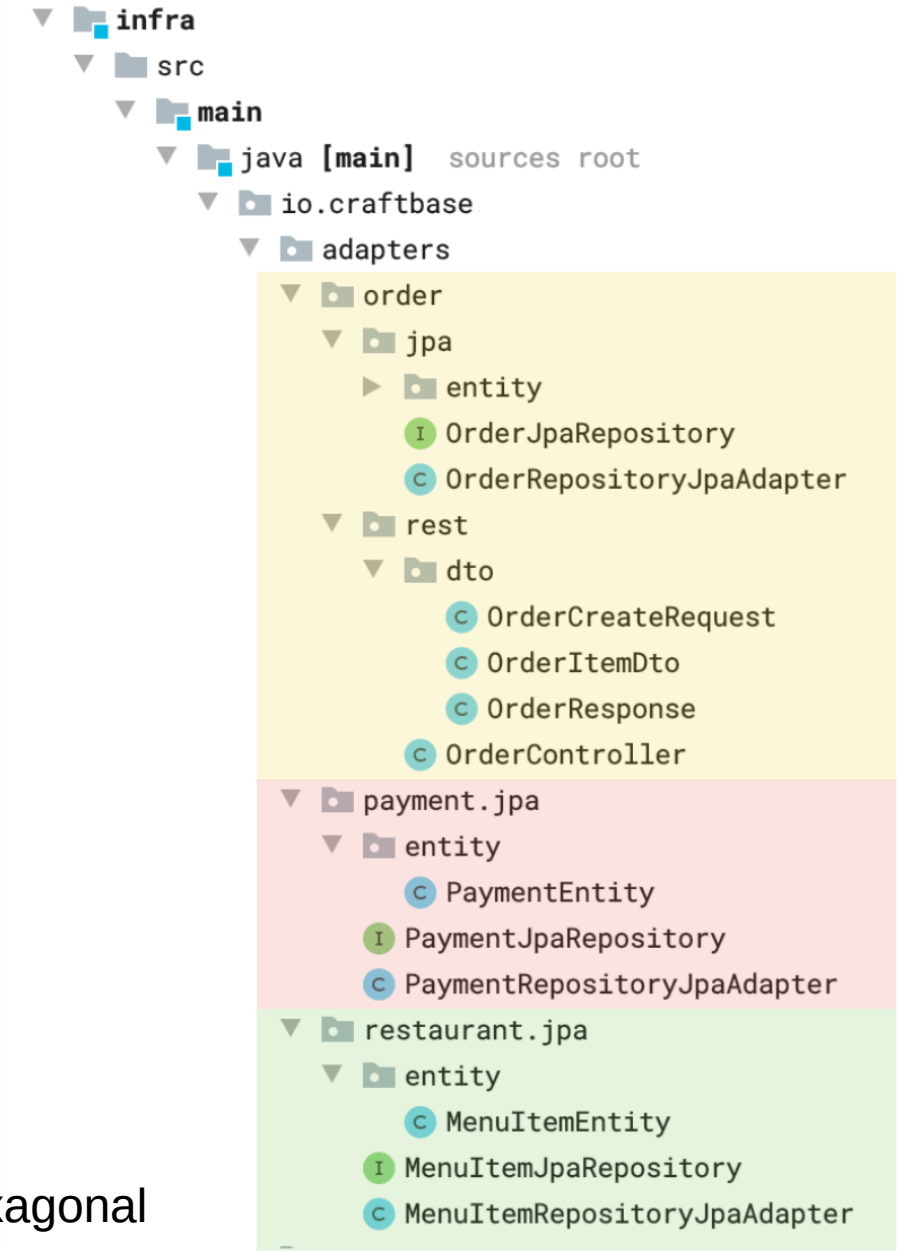
Paternat: Arkitektures softuerik...

- Arkitektura Hexagonale:
  - Arkitektura Hexagonael është një alternativë ndaj paternes arkitektures shtresore.
  - Kjo patern i arkitekturës organizon logjiken e biznesit në qendër.
  - Emërtimet alternative të kësaj paterne: Ports & Adapters/Clean architecture/onion architecture







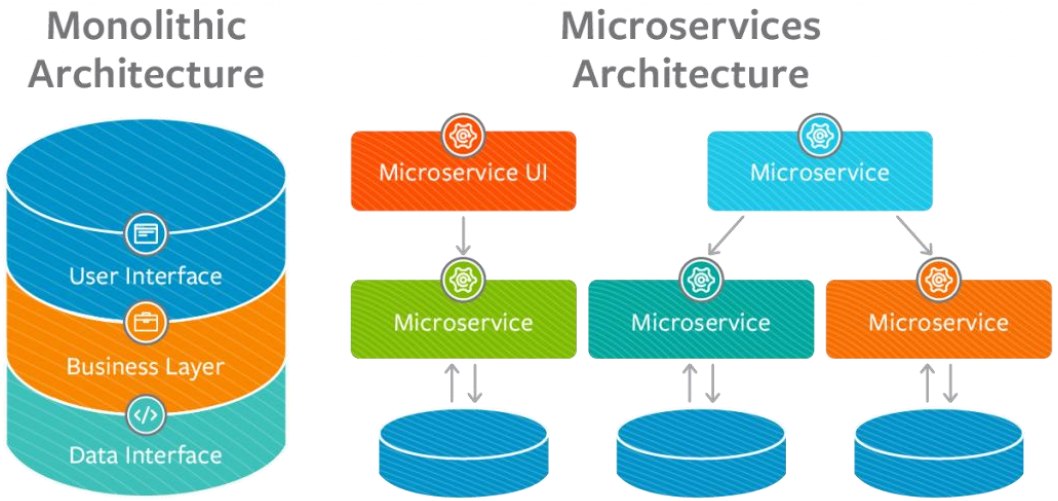


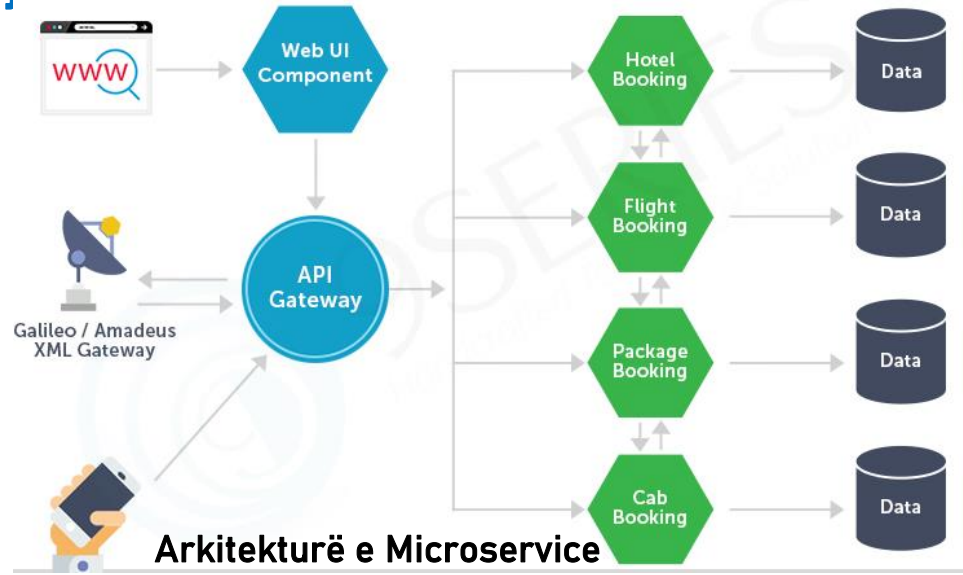
Struktura e organizimit të Kodit në Hexagonal

Paternat: Arkitektures softuerike...

Arkitektura Mikroshërbimeve

- Një aplikacion **monolit** është **një njësi e vetme e përbashkët**, kurse një arkitekturë **mikroshërbimeve**, ku aplikacioni është të zbërthyera në koleksion të njësive më të vogla të pavarura.
- Modulet e Zbatuara (deployable) që komunikojnë me **njëri-tjetrin përmes metodave** të definuar të **quajtura API** (Ndërfaqet e Programimit të Aplikimit).



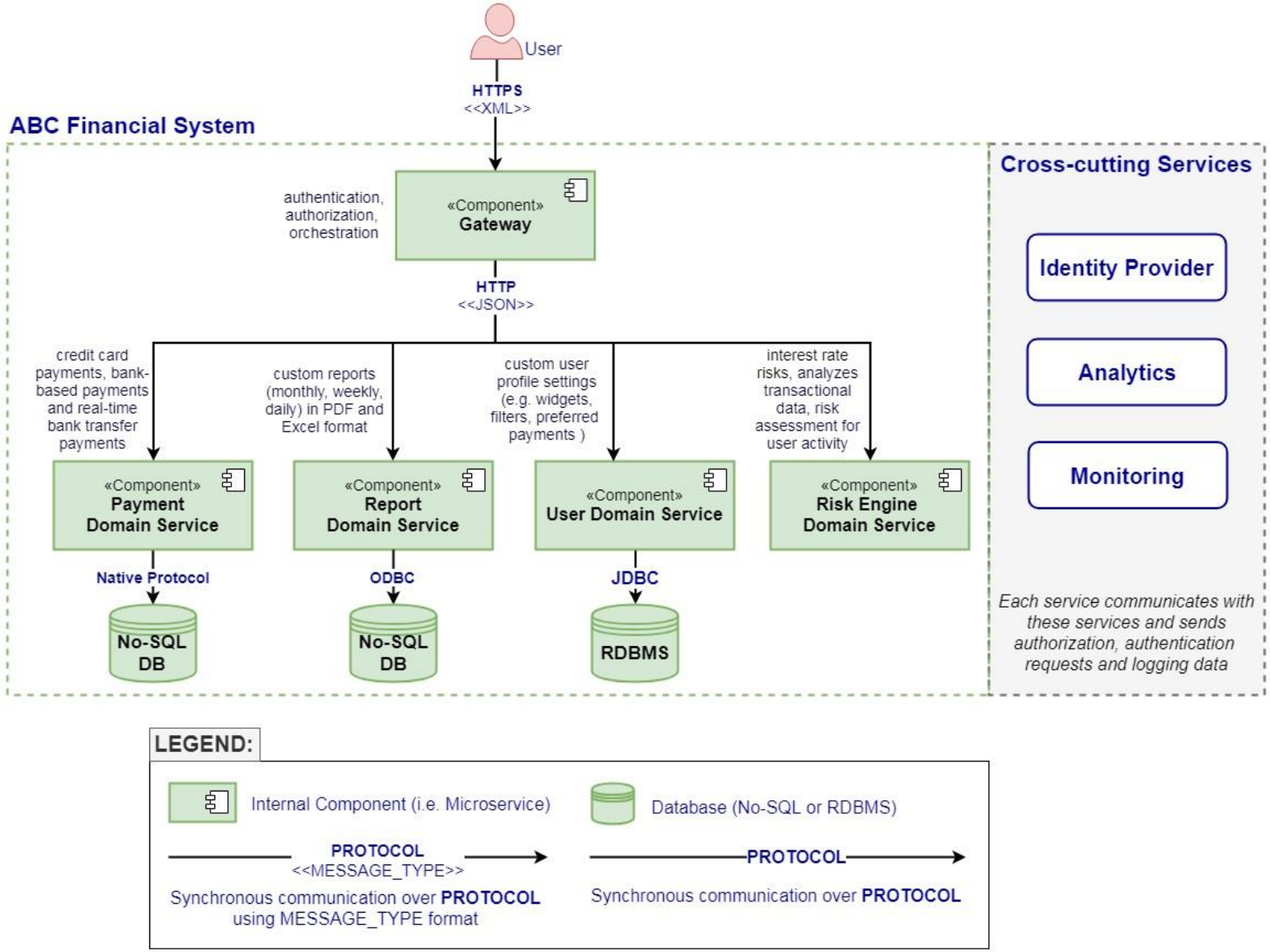


Arkitekturë e Microservice

Shembulli:

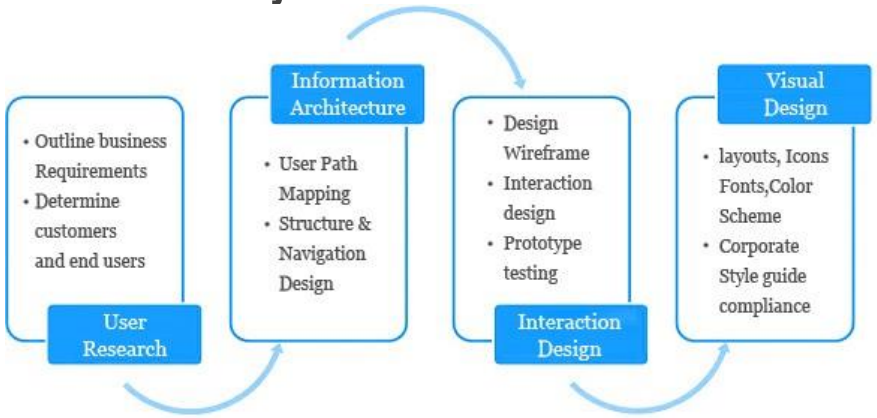
daigramit të komponenteve

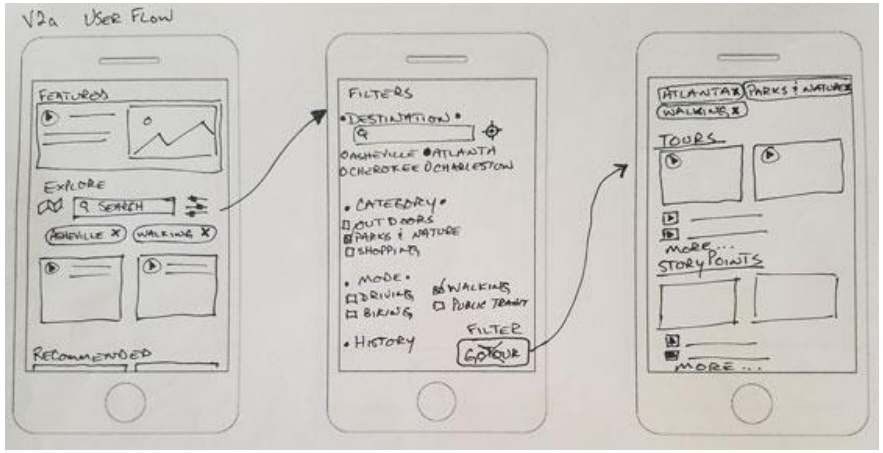
UML-Diagrami i Komponenteve përshkruan që Arkitekturen microservice **e aplikacionit**



Dizajnimi i GUI së Përdoruesit

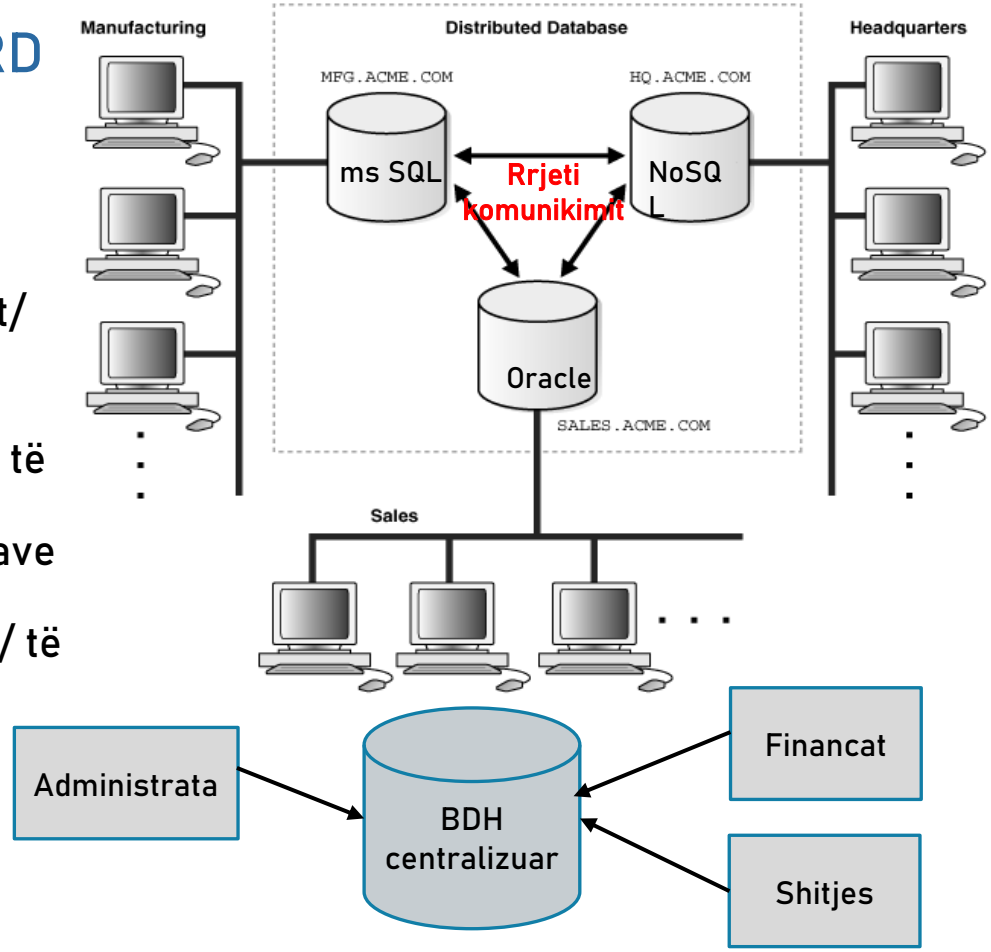
- **Dizajni i ndërfaqes së përdoruesit** ose dizajni **UI** në përgjithësi i referohet paraqitjes vizuale të elementeve me të cilët një përdorues mund të bashkëveprojë në një aplikacion, ose produkt teknologjik.
  - "për përdoruesin, intrerface është sistemi!!"
- **UI** mund të jetë **butonat**, lista **drop-down**, ose **shtrirja vizuale** e një faqe në Ueb.
- Modelet e **ndërfaqes së përdoruesit** jo vetëm që duhet të jenë **tërheqëse** për përdoruesit e mundshëm, por duhet të jenë **funksionale** dhe të krijuara me përdoruesit në mendje.





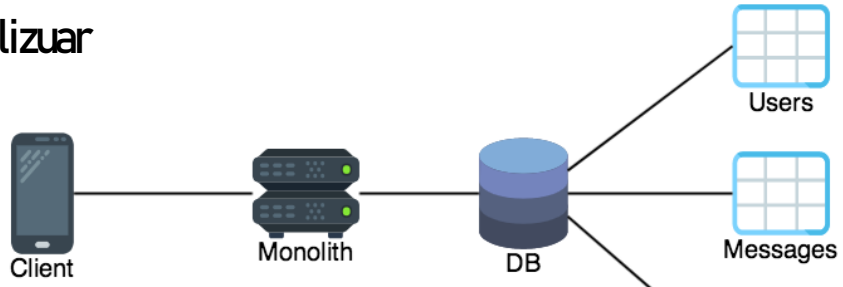
Dizajnimi i Bazës së të Dhënës

- Përdorimi i modelit të domainit të klases ose ERD
- Arkitektura e *Bazës së të dhënave*
  - **BDH centralizuar**
    - + Më e lehtë për të organizuar, modifikuar, kërkuar dhe backups
    - - Mund të jetë më i ngadalshëm për shkak të shfrytëzimit/ ngarkesë të lartë
  - **BDH shperndar ose distributuar**
    - + Qasja e të dhënave dhe kërkimi më shpejtë për system të madhë me ngargesa të larta.
    - - Mund të jetë më i ngadalshëm në përdorimin e të dhënave jo lokale
    - - Duhet të siguroheni që të dhënat janë të qëndrueshme / të sinkronizuara
- Skemat
  - Tabelat dhe kolonat në relacion
- Kufizimite i integritetit
  - Referencat për çelësat e huaj - për tabela lidhëse



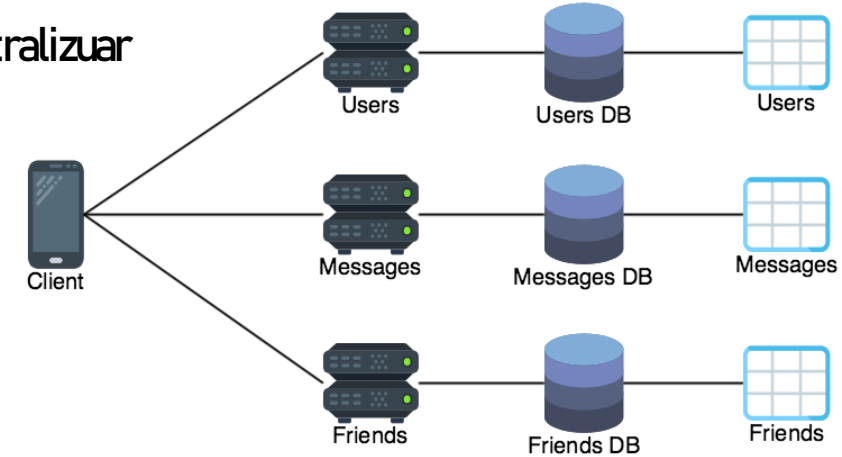
Shembull: Menaxhimi i Decentralizuar i të Dhënave

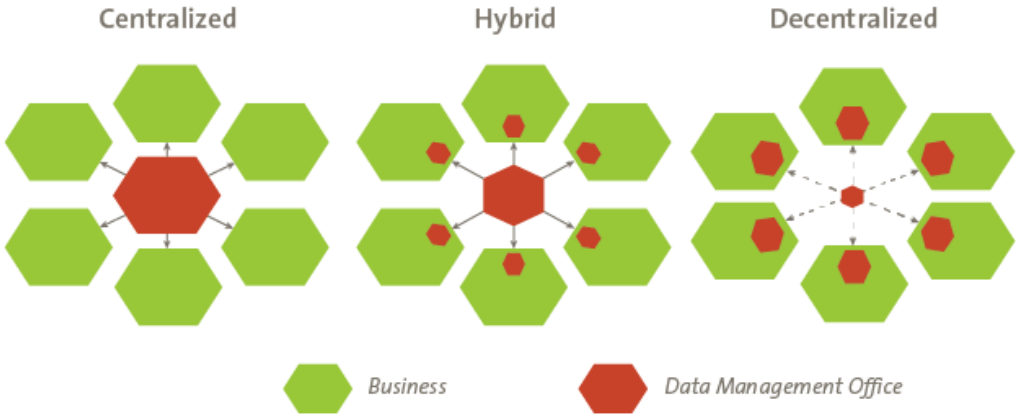
Baza e të Dhënave e Centralizuar



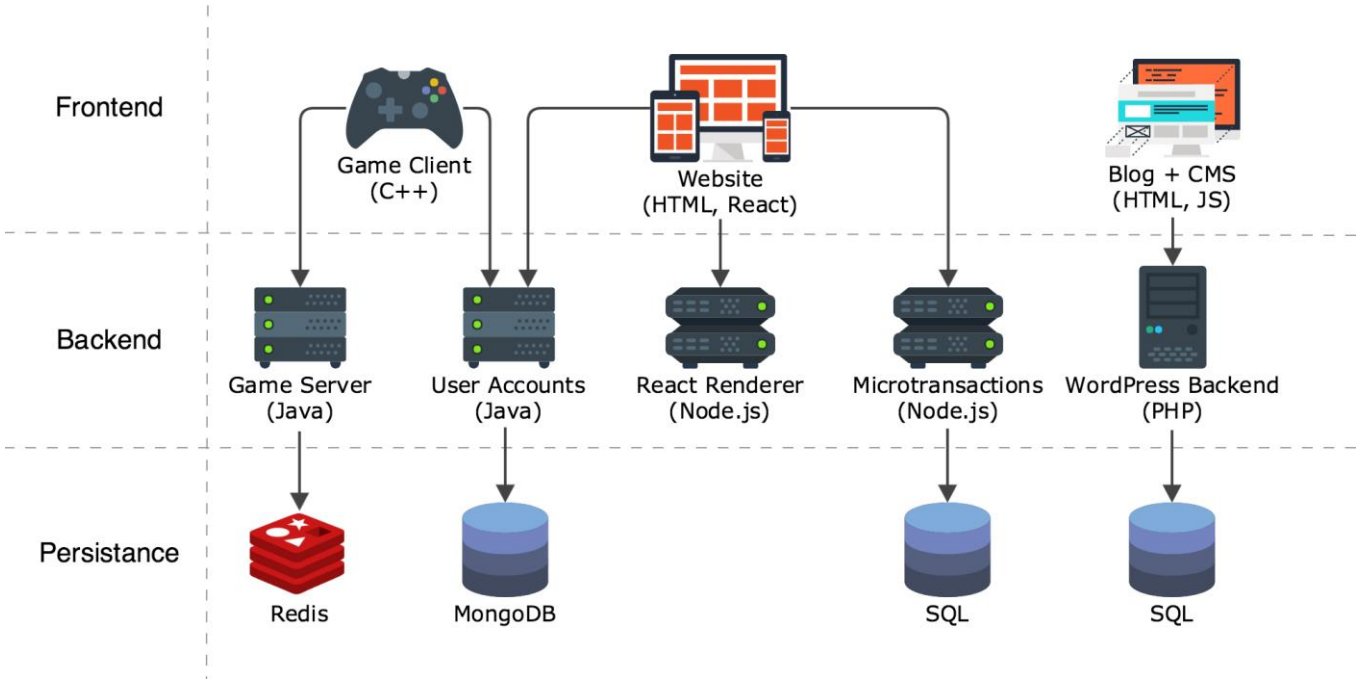
VS

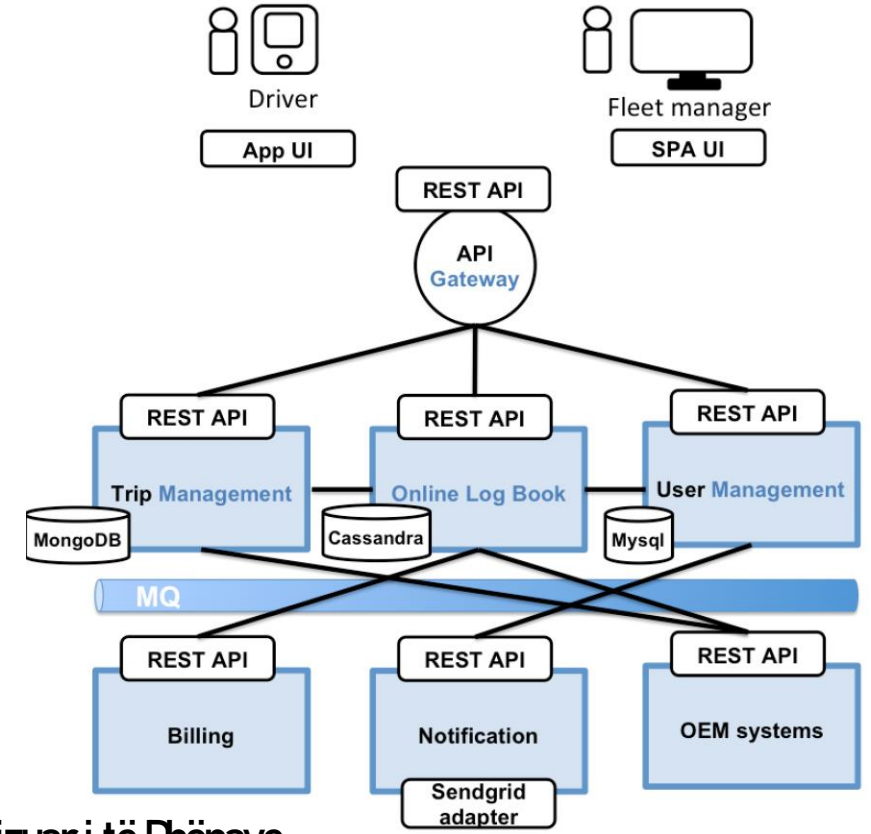
Baza e të Dhënave e Decentralizuar





Shembull: Menaxhimi i Decentralizuar i të Dhënave...

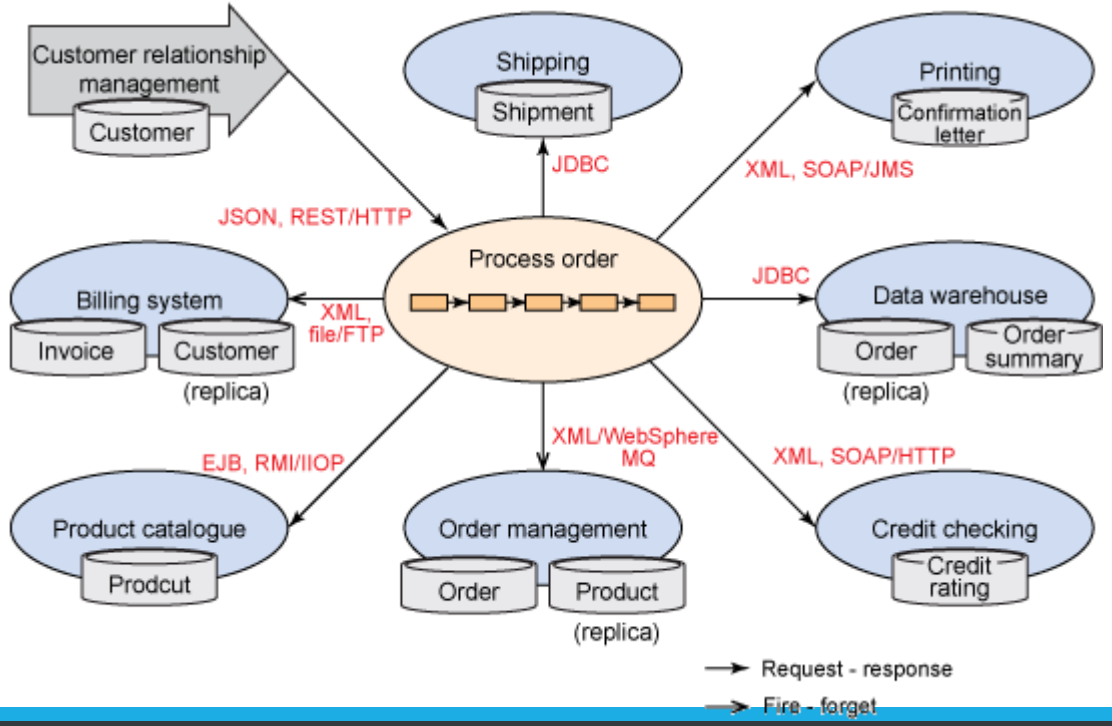




Parimet e mikro-shërbimeve: Menaxhimi i Decentralizuar i të Dhënave

Dizajni i interfaces/nderkomunikimit së sistemit

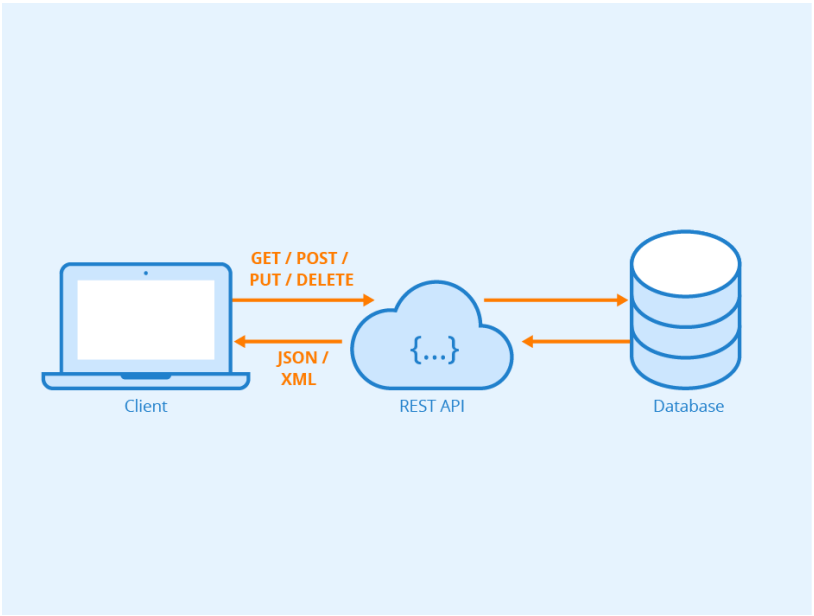
- **Interfaca (API) e sistemit** përpunojnë inputet, *bashkëveprojnë* me sistemet të *tjera* në kohë reale.
  - Kështu që sistemet e tjera mund të flasin me njëri-tjetrin



- Interfesat e sistemit lidhen me sisteme të tjera në mënyra të ndryshme
  - ✓ Ruani të dhënat nga nje sistemi tjeter i përdor.
  - ✓ Lexoni të dhënat e një sistemi tjetër
  - ✓ Bëne kërkesa në kohe reale për informacion.
- Shërbime të softuerike (SaaS)

Shembull: Interface e Sistemit XML/JSON

```
<inventoryRecord>
            <productItem>WS39448-7</productIt
            <inventoryItem>48763920</inventor
            <itemCharacteristics>
                <size>large</size>
                <color>blue</color>
                <options>withzippers</options
            </itemCharacteristics>
            <orderRules>
                <quantityOnHand>54</quantityO
                <averageCost>38.27</averageCo
                <reorderQuantity>25</reorderQ
            </orderRules>
            <dates>
                <dateLastOrder>06042012</date
                <dateLastShipment>08072012</d
            </dates>
</inventoryRecord>
```



tutorial-azure.sql ● {} employees.json × Untitled-1

```
1  [
2    {
3      "EmployeesId": "1",
4      "Name": "Jared",
5      "Location": "Australia"
6    },
7    {
8      "EmployeesId": "2",
9      "Name": "Nikita",
10     "Location": "India"
11   },
12   {
13     "EmployeesId": "3",
14     "Name": "Tom",
15     "Location": "Germany"
16   },
17   {
18     "EmployeesId": "4",
19     "Name": "Jake",
20     "Location": "United States"
21   }
22 ]
```

Application program interface (API) –
The set of public methods that are available to the outside world

Dizajnimi i siguris dhe kontrollit të sistemit

- Kontrolli i ndërfaqes së përdoruesit
  - Autorizimet e përdoruesit
- Kontrolli në nivel të aplikacionit
  - Transaksionet janë "atomike“
- Kontrollet në bazë të të dhënave
  - Nuk ka anomali të bazës së të dhënave
- Kontrollet në Rjet
  - Firewalls, qasjet

Dizajni i aplikacionit/programit (Moduleve)

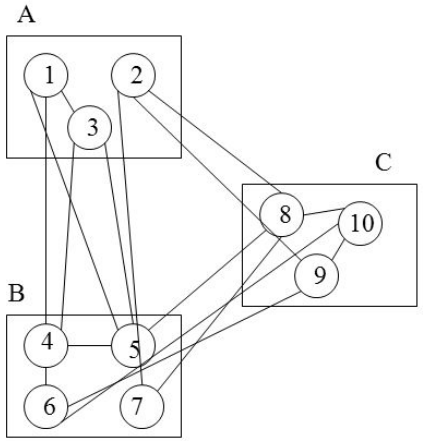
- Dizajni Modular (Modulariteti)-Cili është qëllimi i dizajnit modular të softuerit?
  - Dizajnimi modular i softuerit ka të bëjë me ndarjen e funksionaliteteve në pjesë të pavarura – module.
  - Këto **module** duhet të jenë të **lehta** për tu **kuptuar**, **testuar**, **modifikuar**, **zëvendësuar** ose **fshirë** në mënyrë të **izoluar** pa bërë ndonjë **ndikim të madh** në **pjesën tjetër të sistemit**.
- Çfarë është një modul?
  - Një **modul** është një **pjesë e funksionesh të grupuar** që ekspozohe me një **interface**, me anë të së cilës **modulet** e tjerë mund të **komunikojnë** me të.
  - Pastaj **këto pjesë të vogla** të krijuar në **mënyrë të pavarur** të cilat më vonë **do të kombinohen** në një **sistem më të madh**.
  - Ju gjithashtu mund atë të përshkruarajm si një **komponent** ose një **bounded context** (kontekst i kufizuar).
  - Për shkak të **ripërdorimit** të tij, dizajni modular është shumë i **dobishëm** dhe i **qëndrueshëm**.

Dizajni i aplikacionit/programit (Moduleve)...

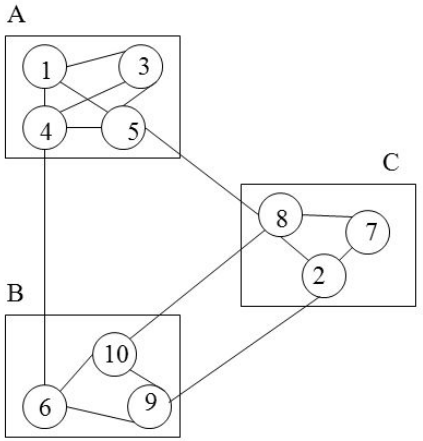
- Modulariteti/Modulizimi:
  - Procesi i **zbërthimit të një softueri** në **shumë module të pavarura** ku secili modul zhvillohet veçmas quhet **Modulizimi**.
- Në mënyrë që të ndërtohet një softuer me dizajn modular efektiv ekziston një faktor "Pavarësia Funksionale" që hyn në lojë.
  - **Pavarësia Funksionale** është se një **funksion-atomike** ai kryen një **detyrë të vetme** të softuerit pa ose me **ndërveprimin të vogël** me **modulet** e **tjera**.
- Pavarësia e moduleve mund të matet duke përdorur 2 kritere:
  - Cohesion dhe Coupling

Dizajni i aplikacionit/programit (Moduleve)...

- Cohesion vs Coupling:
  - **Cohesion** ka të bëjë me atë se sa elementet brenda një moduli veproj ose përkasin së bashku dhe i shërbejnë një qëllimi të përbashkët.
  - **Coupling** ka të bëjë me atë se sa një modul varet ose ndërvepron me modulet e tjerë.



Bad modularization:
low cohesion, high coupling



Good modularization:
high cohesion, low coupling

Secili modul duhet të **implementohet** në një mënyrë që të ketë **më pak varësi (less dependency)** nga **modulet e tjera** dhe
Elementet brenda këtij **moduli** duhet të **lidhen funksionalisht së bashku.**

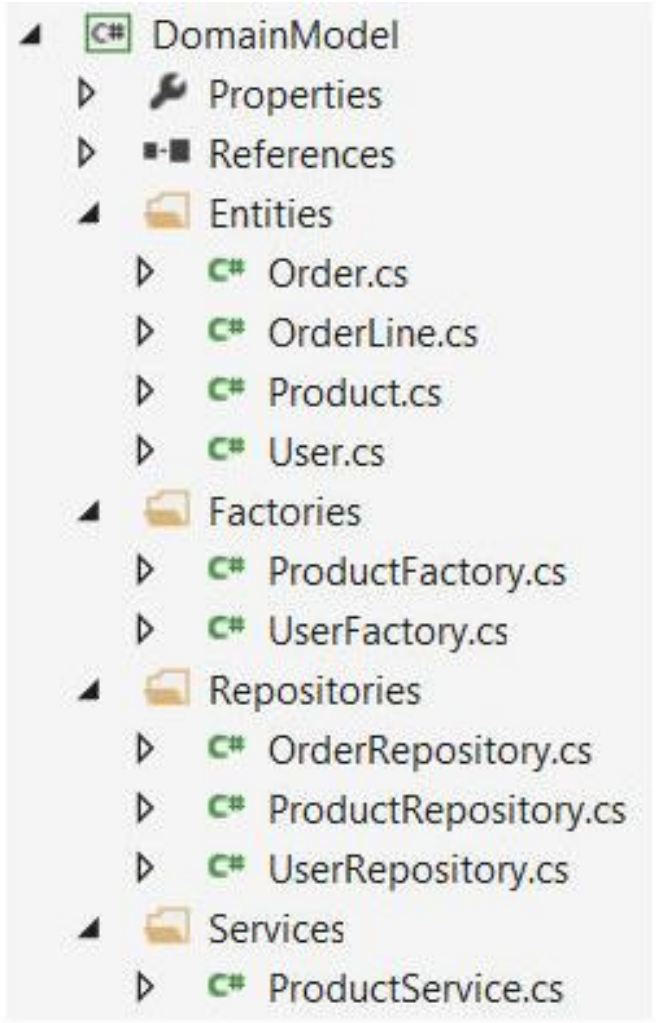
Një **dizajn i mirë** i softuerit **kërkon kohezion të lartë** dhe **coupling të ulët.**

Shembull:

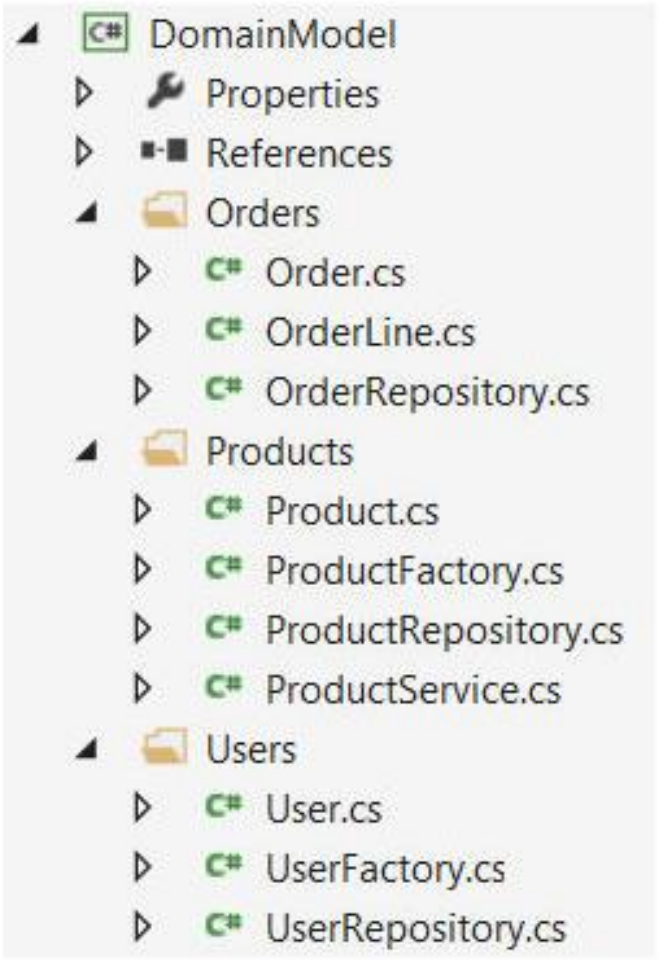
Organizimit të projektit sipas

Parimeve Cohesionit dhe Coupling

Shembull:
Organizimit të projektit sipas parimeve Cohesionit dhe Coupling



(a) Low cohesion, high coupling

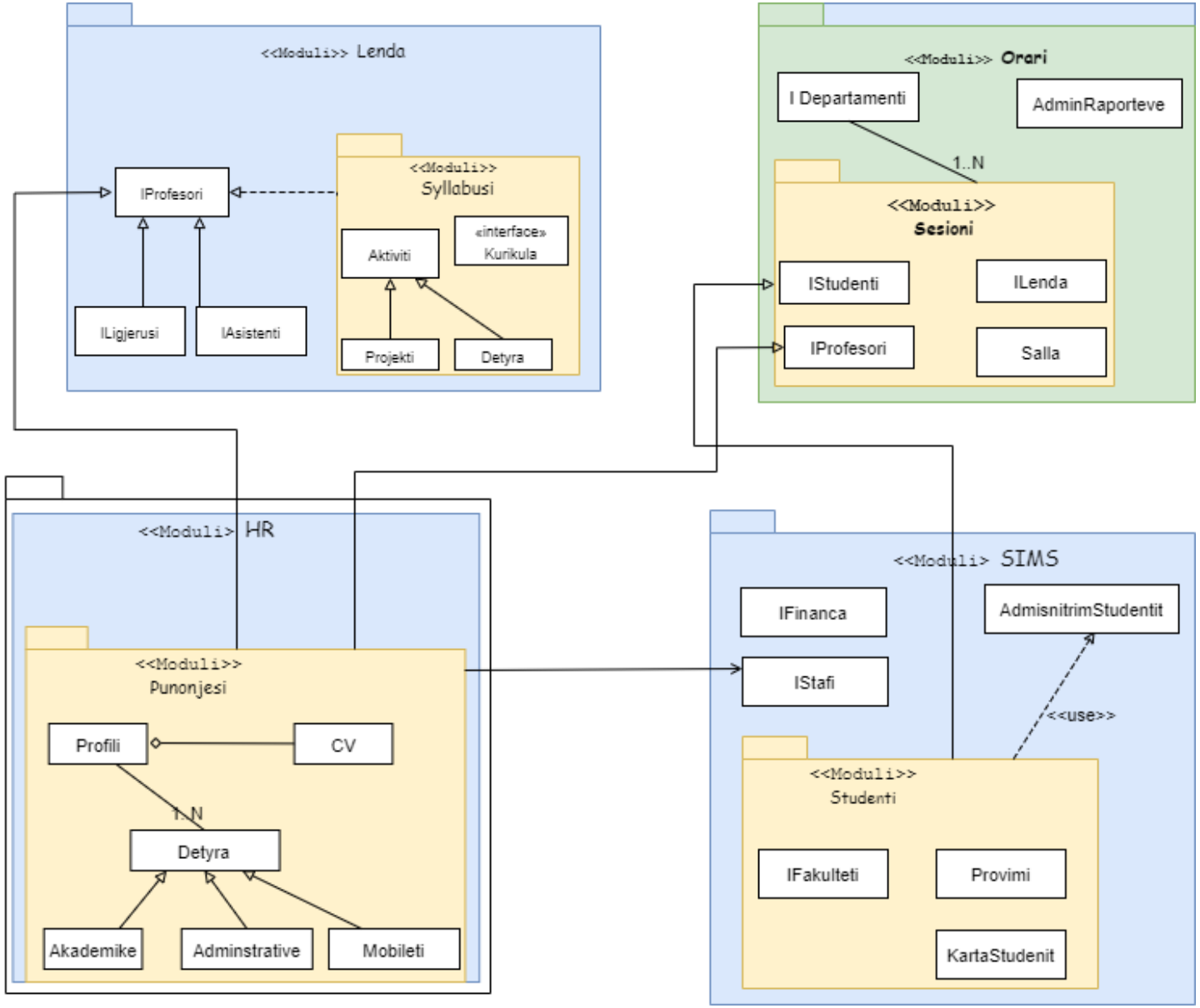


(b) High cohesion, low coupling

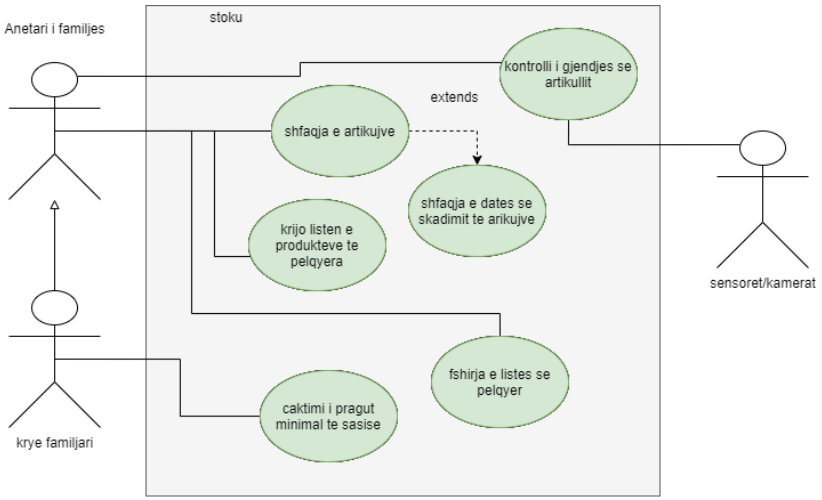
p.sh: Pamja Logjike

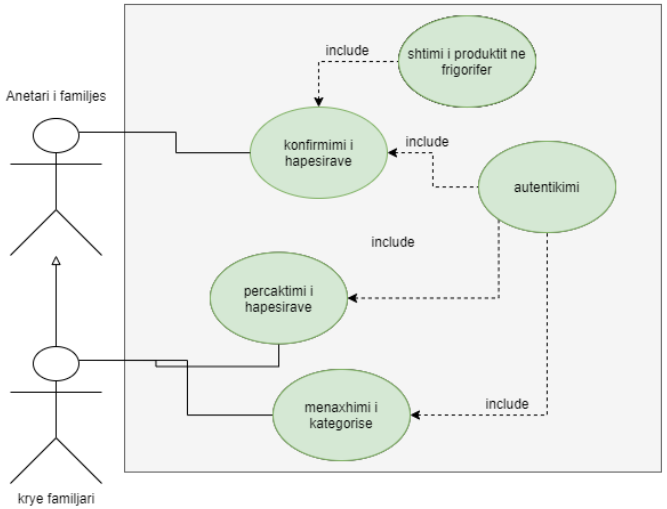
Modulet

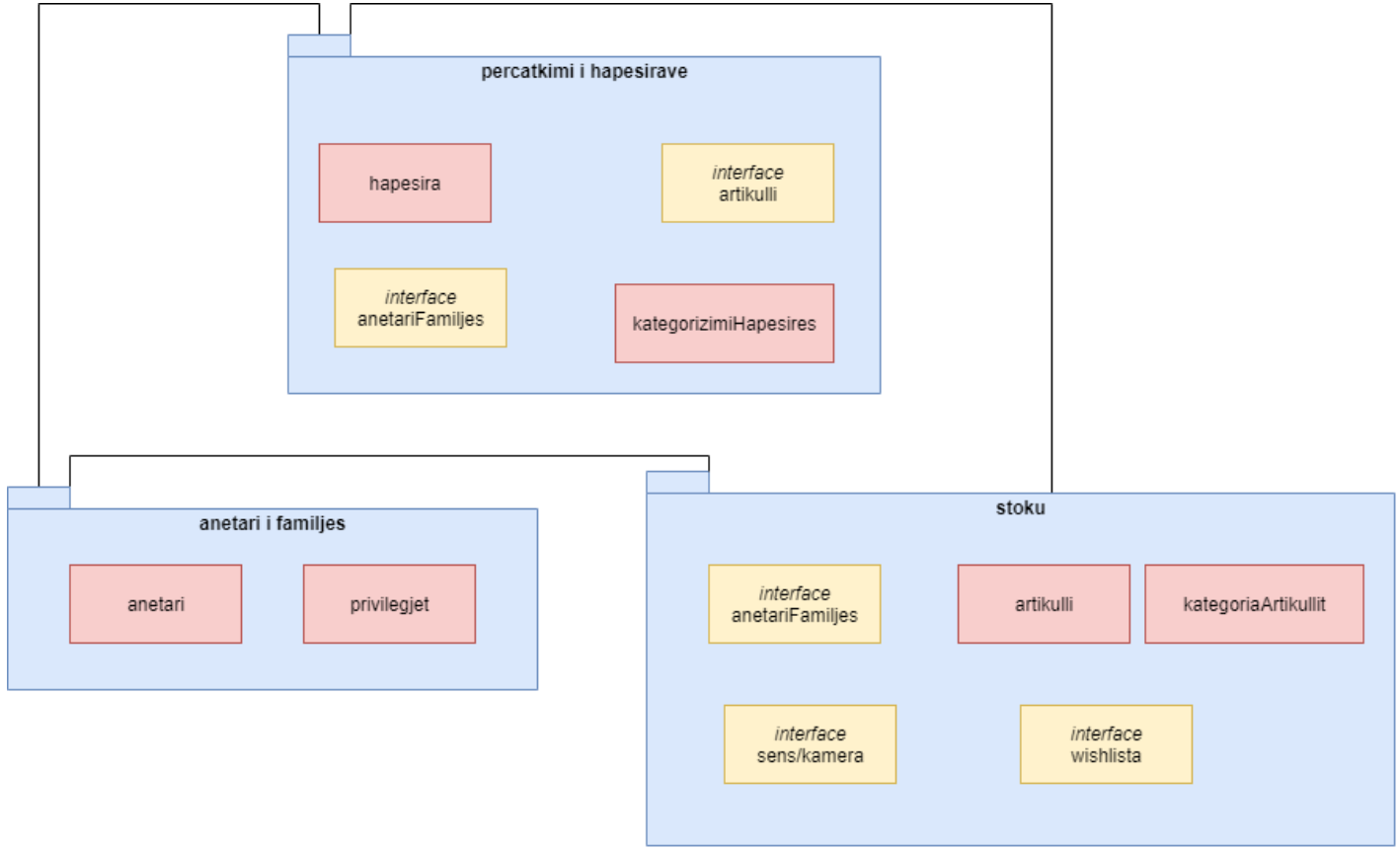
shumbulli: rasti regjistrimi i **studentëve në orarin** e Ligjeratave

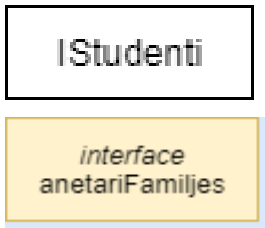


Dizajni i disa Moduleve për sistemin **SmartFridge**









- p.sh *Interface antariFamiljes/IStudenti* d.m.th me simbolin referohemi interface se Entiteteve në vend të moduleve (entiteteve) konkrete
- nënkupton klasat/objektet /entitetet/ që implementojnë interface
- Nuk është një interface mbrenda Modulit përkates

Për t'i qasur **metodatve e interfaces**, interfaca duhet të "*implementohet*" nga një klasë tjetër me fjalën kyçe **`implements`** dhe metodat duhet të implementohen në klasën e cila trashëgon vetitë e interfaces.

Pyetje

Faleminderit...!